DOCUMENT RESUME

ED 211 066 IR 009 885

AUTHOR Ward, Lawrence M.; Steiger, James A.
TITLE The TRS-80 Microcomputer Network as the Foundation of a Computer-Based Instructional Laboratory.
PUB DATE Jun 81
NOTE 13p.; Paper presented at the Annual Meeting of the Canadian Psychological Association (Toronto, Canada, June 1981).

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; Computer Graphics; *Computer Programs; Foreign Countries; Higher Education; *Learning Laboratories; *Microcomputers; *Psychology; Simulation
IDENTIFIERS *TRS 80; *University of British Columbia (Canada)

ABSTRACT

The basic hardware configuration of a network of up to 17 TRS-80 microcomputer systems (1 central, 16 peripheral) is described as implemented in the psychology department at the University of British Columbia. Such a system can serve several instructional functions (e.g., data generation for lab experiments, simulation of theories or experiments, hands-on practice in independent research, interactive graphics display of difficult concepts) in a wide variety of psychology courses, including introductory, experimental, quantitative methods, perception, cognition, and advanced lab courses. It can function either as a block-scheduled research experience for large courses, as a drop-in research utility for advanced lab courses, or as a utility for data collection in course projects in research methods or lab courses at both undergraduate and graduate levels. A wide variety of software is already available for this system, and other software can be readily adapted or developed out of computer programs used for research or lab exercises. Some disadvantages in comparison to other systems include weak graphics and a lack of concern about upward compatibility of older devices with newer products. Several recommended books and utilities are listed, and six figures, including a diagram of the network, are appended. (Author/MER)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED211066

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

X This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

THE TRS-80 MICROCOMPUTER NETWORK AS THE FOUNDATION OF A COMPUTER-BASED INSTRUCTIONAL LABORATORY

by

Lawrence M. Ward
and
James A. Steiger

IR009885

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Lawrence M. Ward

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Canadian Psychological Association, June, 1981, Toronto, Ontario

The TRS-80 microcomputer network as the foundation of a computer-based instructional laboratory

Lawrence M. Ward and James A. Steiger
University of British Columbia

1. HARDWARE

At UBC we have implemented a TRS-80 Model I microcomputer network as an instructional laboratory for a variety of psychology courses. The network (Figure 1) consists of one central system and up to 16 satellite systems (we have 10 at present). The central system consists of a TRS-80 Model I microcomputer and keyboard with 16 Kb of memory, an expansion interface with 16 Kb of additional core, disk controller, and printer interface, two 5" floppy disk drives, a quick printer, a video monitor, a cassette tape recorder, and the Network I controller. Each satellite system consists of a TRS-80 Model I microcomputer with keyboard and 16 Kb of memory, a video monitor, and a cassette tape recorder. The network controller input is through the cassette port, so both cassette and network controller cannot be simultaneously active. Total cost of this system was about $16,000 in March 1980 from Radio Shack in Vancouver.

The Network I controller has limited but significant capabilities. It can distribute the same program (file) to any subset of the up to 16 satellite stations connected to it by dumping it in parallel from the memory of the main computer to the memories of the selected satellites via their cassette inputs. This is a rather slow process (500 baud) but since it is done in parallel it is acceptable. A typical experimental program load would take from 1 to 5 minutes (central to satellites) and anywhere from a few seconds (if program is on disk) to 1-5 minutes (if program is on cassette) to load into the central system. Satellite stations can also send their programs (files) to the central computer, one at a time, where they can be stored on disk or tape, printed, or sent out to a different satellite. Unfortunately, both the send and receive processes require cooperation of both satellite(s) and central system; thus no control of central system over satellites is possible, and the satellites do not have access to disk or printer unless the central system is set to accept the input. Newer models should improve on these limited networking capabilities but at some additional cost.

2. USES

At present we use the lab to run demonstrations of psychological concepts that are difficult to get a feeling for in a lecture context (eg., effects of various manipulations on d' and β, feature scanning) in lower level courses. We use the lab for data collection in laboratory courses (eg., effects of caffeine on reaction times, illusion decrement, signal detection theory). The computers are also used for individual research projects and for demonstrations of difficult statistical concepts (eg., central limit theorem, law of large numbers) in the research methods course. Finally, next year the lab will be used by students in a lab course in Computers in Psychology. In this course students will begin to use the full capability of the network and the computers themselves as they learn to program them to run experiments and data analyses, teach, simulate, and even do clinical interviews.

We use a mixture of block scheduling (for scheduled lab sections) and language-lab-like drop-in use during unscheduled time. Our storeroom technician has been trained in basic network operation and now acts as a language-lab

supervisor would, setting up students who drop in for extra or makeup work with the appropriate program(s) on satellite systems and maintaining system security.

3. SOFTWARE AVAILABILITY

There is enough reasonable software commercially available that a lab like ours can be functioning soon after purchase. Life Science Associates (One Fenimore Road, Bayport, New York, 11705) sells (for $300) a package programmed by Thomas Perrera that is supplied on cassette tapes (but, unfortunately, not on disk). The package includes programs/demonstrations dealing with signal detection theory, psychophysical threshold methods, paired associate learning, visual illusions, reaction time, and others, plus a program that allows the computer to be used as a tachistoscope and one that simulates several models of the effects of reinforcement schedules on operant conditioning. In the latter, even the subject (a simulated rat who presses a simulated bar) is supplied (Figure 2). These programs typically involve running the student in an experiment (with variable parameter values) and then allowing a choice of data analyses. The programs generally run, but some have bugs (eg., incorrect d's in the signal detection experiment) that we have corrected on our system. The experimental paradigms are adequately implemented, if a trifle old-fashioned, but the programming is often cumbersome and graphics are slow (see later).

Conduit (P.O. Box 388, Iowa City, Iowa, 52244) is marketing (for $145) a package of programs by Fischler, Griggs, Warner, Sherman and Levy called Perception and Cognition Laboratory. This package is of more contemporary interest, including exercises in feature search (Neisser paradigm - Figure 3), iconic memory (Sperling paradigm), memory scanning (Sternberg paradigm), letter-name identification (Posner paradigm), and sentence-picture verification - as well as sigral detection and difference threshold measurement. Further it is distributed on 5" diskette. However, it runs without adaptation only on the central, disk-oriented, system. We had to rewrite the various programs more or less extensively to get them to run on the satellite systems. Typically the timers (which used the 25 msec "heartbeat" timers of the expansion interface) had to be changed to a software timing loop, since the Level II 16 Kb machine doesn't have the hardware, any uses of Disk BASIC changed to Level II BASIC, and the data display changed from printer to videoscreen format. The programs also have drawbacks. The programming is clumsy at times and the use of a central monitor program (with a fancy logo that obviously consumed programming time better spent on the exercises) that loads and starts the exercises is clumsy for a network system. We had to reprogram individual programs and store them separately for separate loading. The data analyses are fixed and are not as extensive as with the Perrera package; in some cases they are non-existent: eg., the Posner paradigm program prints out 128 trials of raw data (3 data per trial).

Extensive software support is also available for the disk system itself, eg., several disk operating systems, Fortran and Basic compilers, Editor/Assembler, a Text Processor, a Forth compiler. However, all programs for use in the network must ultimately be Level II BASIC source programs or compiled programs (object programs) that will run on the Level II Model I machine. None of the power of the central system is available to the satellites in this primitive network.

4. SOFTWARE DEVELOPMENT

Of course it is possible to write one's own lab exercises. We have implemented several. Some were written specifically for the lab (eg. interactive demonstrations of the central limit theorem - Figure 4), some were adapted from previous lab software (eg. a simulation of the Limulus compound eye for use in a race for the Hartline Prize - Figure 5), and others were adapted from research programs (eg. a 2-D mental rotation paradigm - Figure 6). However, a number of factors limit the speed and effectiveness with which reliable software can be

developed for a proposed laboratory exercise. These include rather serious limitations in the TRS-80 onboard graphics hardware and inadequate speed of the TRS-80 BASIC language in timing and high-speed graphics applications. We discuss these limitations and some possible "cures" below.

4.1. TRS-80 graphics. The TRS-80 computer has a graphics capability that is rather limited. First, of course, color is not implemented. Perhaps more important, the ultimate resolution of the graphics is extremely coarse. The screen is divided into 1024 (16 x 64) graphics blocks, each containing 6 rectangular elements. These elements can be set on or off independently, either by using "SET" and "RESET" commands from BASIC, or by using the "POKE" command to place graphics control codes directly in the screen memory locations.

The low resolution problems with the graphics system are compounded by the use of rectangular graphics blocks (instead of square ones). In general, the user will find it very frustrating trying to develop sophisticated graphics with this system.

The TRS-80 (as far as we know) lacks the capability for software suppression of the video display. Hence, it is not possible to use hardware to (1) suppress the display, (2) create a (hidden) picture, and then (3) present it almost instantaneously by turning the display back on. A software solution using machine language programming is available (see 4.2 below). However, the BASIC interpreter is simply too slow to provide fast display presentation on its own.

A (relatively) low-cost ($350) solution to many of the problems with the TRS-80 graphics is a hardware "add-on" system, the "ERAM extended random access high resolution graphics" system, marketed by Keyline Computer Services of Tulsa, Oklahoma. This system can be added to a TRS-80 Model I computer with no hardware modifications to the TRS-80. The unit is simply plugged in between the keyboard and display of the unit. ERAM creates a high-resolution (192 x 256 individually controlled graphics dots) display, which exists in parallel with the ordinary TRS-80 graphics, and which can be set on, off, or reversed by flipping a single control bit. With this system, the user can create a high-resolution graphics display "behind the screen," and then turn it on almost instantaneously. With the ERAM unit turned off, the TRS-80 operates normally. The unit comes with some excellent software support and demonstration programs, and seems to represent very fine value for users who desire extended graphics capabilities.

4.2. Limits of Interpreter BASIC in Timing and Graphics Operations. The standard interpreter BASIC system on the TRS-80 interprets each line of source code into machine language just prior to execution. This system avoids the need for recompiling an entire program when a single small error is detected, but it also reduces execution speed (vs. compiled machine language) by a factor of (roughly) 30. In many applications, this loss of speed is not even noticeable. However, consider a situation in which reaction times to a complex display are to be measured, as in a standard "mental rotation" paradigm. In this case, the display must be presented in a few milliseconds. Interpreter BASIC is, in general, too slow to do this. The easiest software solution to this problem is as follows: First, dedicate 1024 bytes of RAM as a "dummy screen" area. Second, assemble the desired display, using POKE statements from BASIC, in the "dummy screen" memory. Finally, execute a "block move" machine language instruction to transfer the 1024 byte "dummy screen" directly into the 1024 byte video memory. The onset of the screen display appears instantaneous, although it actually takes about 5-10 milliseconds.

The user with advanced applications will eventually find interpreter BASIC, by itself, to be inadequate. One solution is to learn assembly language programming, and use assembly language routines to augment the capabilities of BASIC. This approach is usually adequate in situations where critical problems in execution speed are confined to a few key points in program design.

In some situations, however, deficiencies in speed are not confined to a few areas of the program. This is usually the case where extensive computations must be performed between trials of an experiment, for example. In this case, the user may have to compile BASIC code into assembly language to gain the necessary speed. Several BASIC compilers are being marketed, but only one, ACCEL2, has full compatibility with interpreter BASIC and allows program development within a 16Kb Level II environment. This program, as sold, does not fully utilize memory within a 16Kb system (it was designed for use on a 32Kb disk system). However, the second author has restructured the compiler code for maximum efficiency in a 16Kb configuration. This revised version allows interpreter BASIC programs of length (approximately 12-13Kb to be compiled and run in 16Kb of RAM. The speedup factor in the compiled code varies greatly, but is approximately 5-10 overall, and approaches 30 for operations that extensively utilize integer variables. ACCEL2 is available from Allen Gelder Software, San Francisco, CA.

For development and application of machine language software, the following books and utilities are highly recommended, if not invaluable:

1. Radio Shack Disk Editor-Assembler. This is a remarketed version of the Microsoft assembly language development system. It allows users to create assembly language modules that can be linked and relocated automatically in any desired area of RAM. The assembler also supports macros. If you plan to do much assembly language programming, this program is absolutely necessary. Unfortunately, the documentation is terse and written strictly for the professional programmer.

2. RSM2D Disk Monitor. This set of machine language utilities includes a dissasembler (which translates machine code back into assembly language), debugging monitor, and some excellent utilities for transferring programs onto tape for use in a 16Kb environment. This very useful package is an excellent value at about $50.

3. Books on assembly language programming:

Leventhal, L.A. Z80 assembly language programming. Berkeley, CA: Osborne/McGraw-Hill, 1979.

Leventhal's book is excellent, both as a text and a reference book. It retails for about $20 in Canada, but, in our opinion, it is well worth it. The book is authoritative, clear, and superbly organized. If you're going to buy one book on Z80 assembly language, get this one.

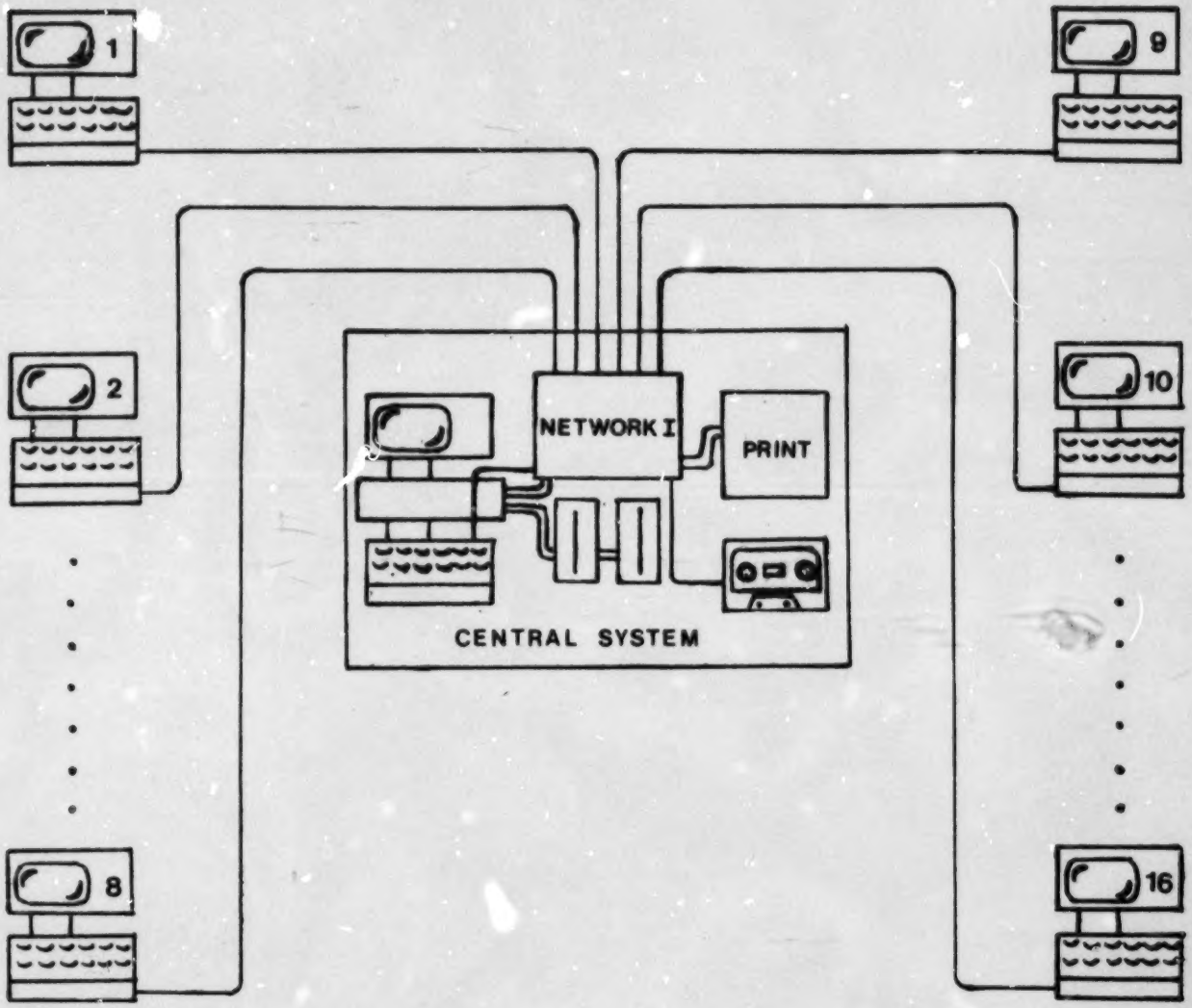
Barden, W.A. TRS-80 assembly language programming. Fort Worth, TX: Radio Shack, 1979.

This book is useful as a very limited introduction. It is inexpensive, (about $5 in Canada), and readily available at Radio Shack stores. It lacks meaningful exercises, and concentrates (for our taste) too much on very basic operations. The reader is left with little or no conception of how to develop a major assembly language project.

Spracklen, K. Z-80 and 8080 assembly language programming. Rochelle Park, N.J.: Hayden, 1979.

This book is well written, and presents 8080 programs side-by-side with those written for the Z80. It has exercises with answers, and is similar in level to Barden's book. It sells for about $8 in Canada.

FIGURE 1



TRS-80 MICROCOMPUTER NETWORK

FIGURE 2

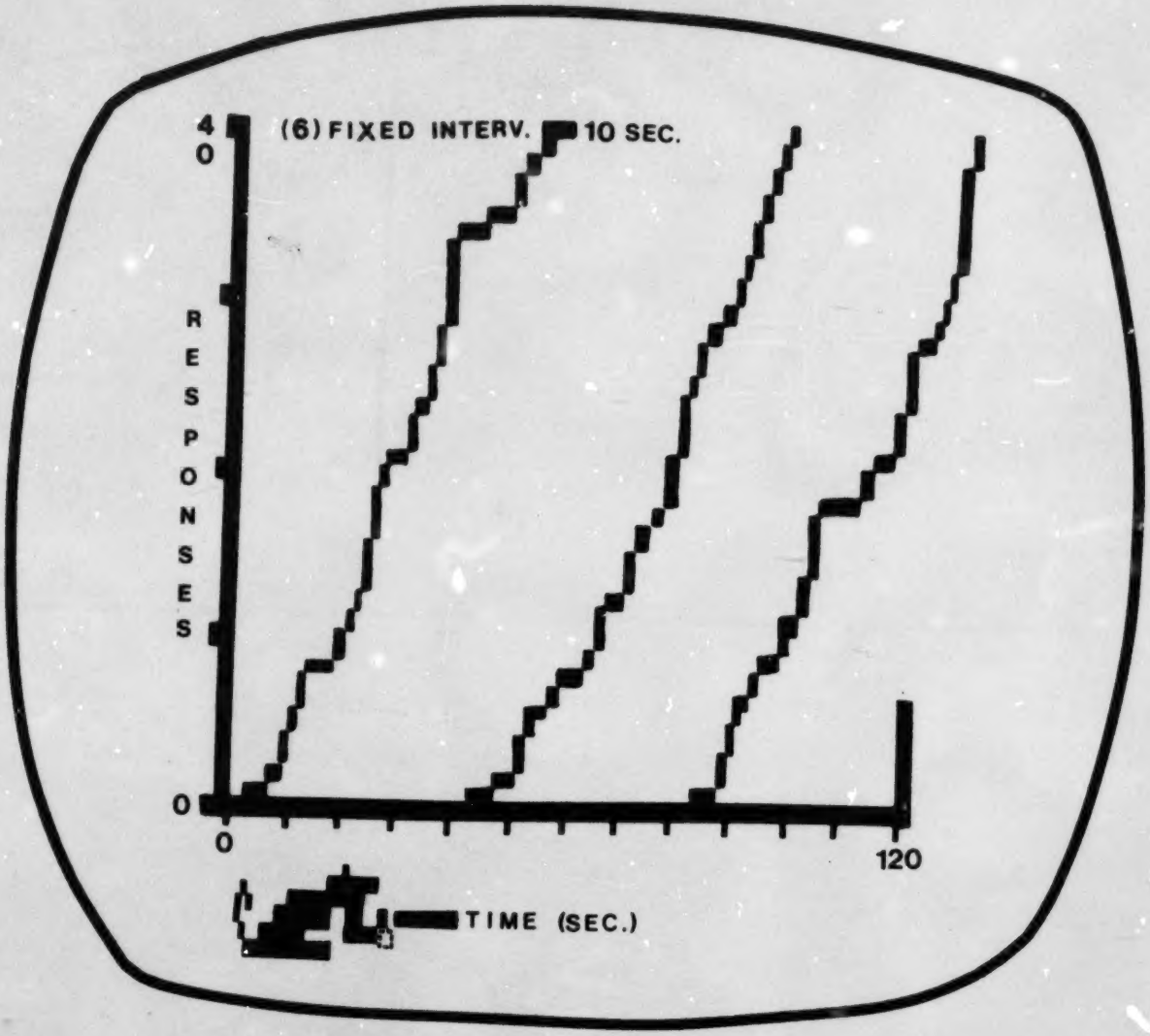


FIGURE 3

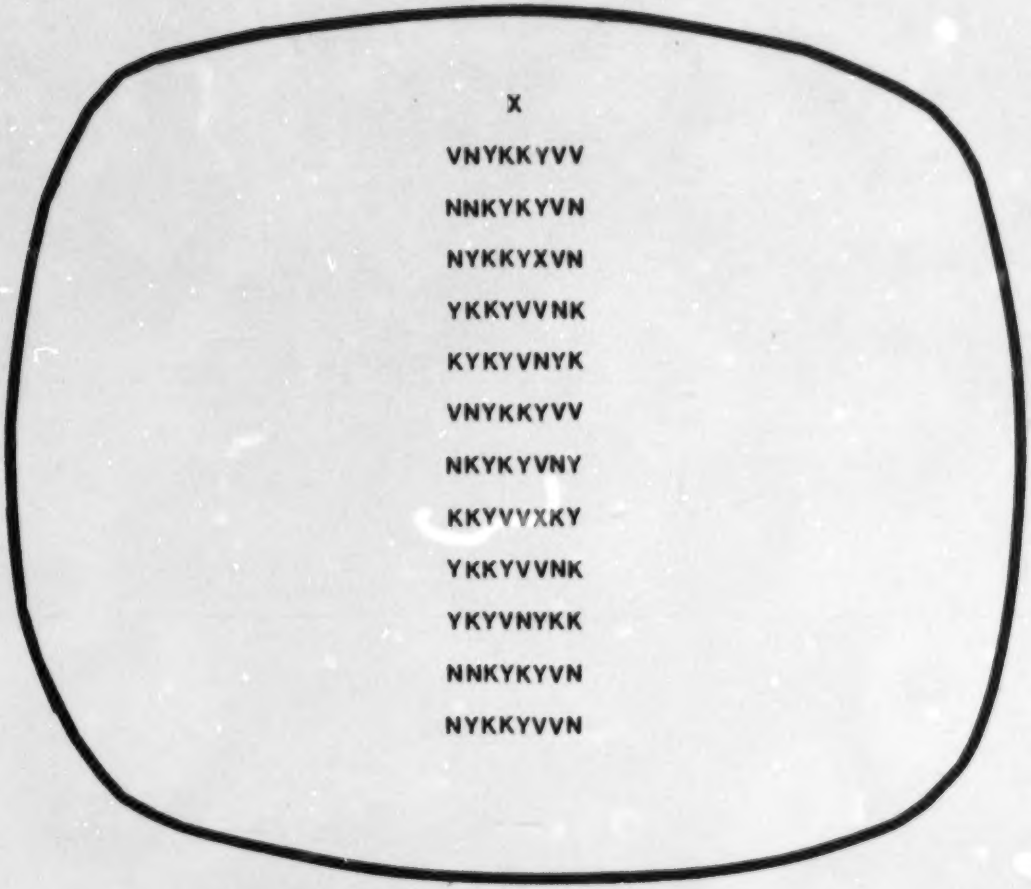
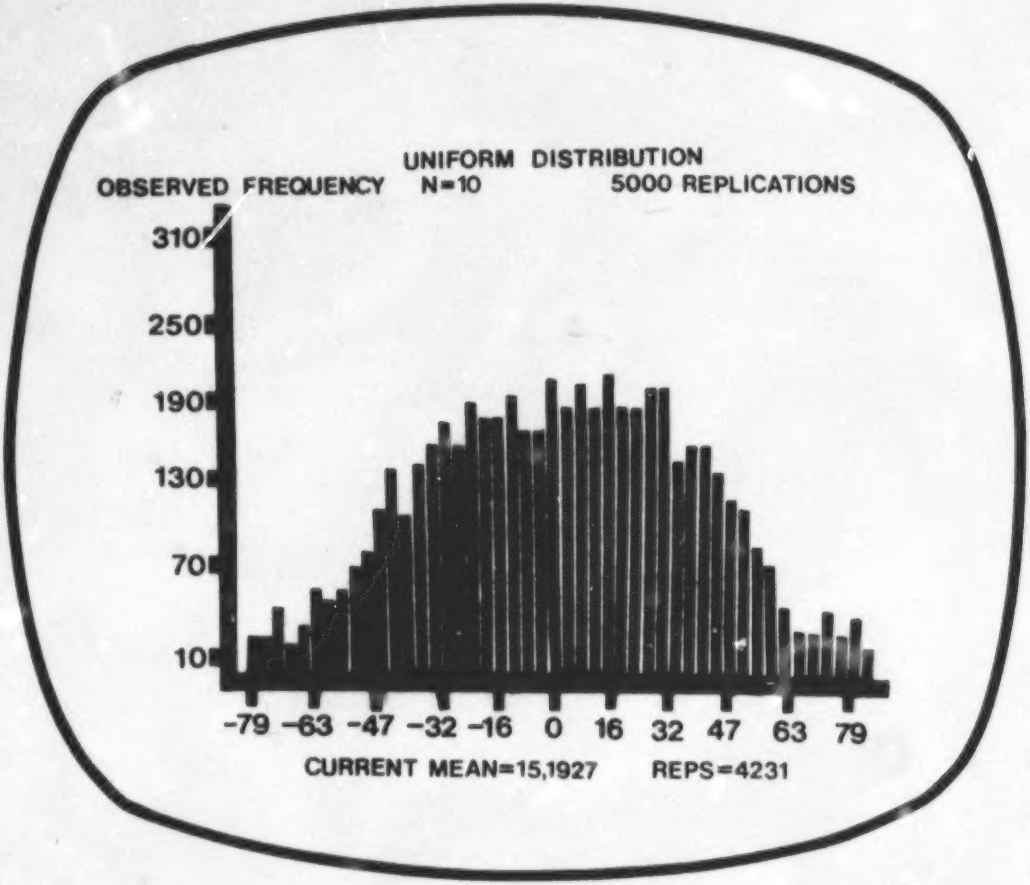

X
VNYKKYVV
NNKYKYVN
NYKKYXVN
YKKYVVNK
KYKYVNYK
VNYKKYVV
NKYKYVNY
KKYVVXKY
YKKYVVNK
YKYVNYKK
NNKYKYVN
NYKKYVVN

FIGURE 4
CENTRAL LIMIT THEOREM



H.K. HARTLINE PRIZE
FOR
SIMULATED EXCELLENCE

Awarded in recognition of excellence and nobel endeavour
in ersatz visual neurophysiology and artificial intelligence

Under the auspices of H.R.H. King Lüss I

Dated this 26 day of January in the year 1981

Limm U. Lüss I
King of Sweden

Rod N. Kones
President,
Awards Committee

FIGURE 6
MENTAL ROTATION TASK